**《C 语言程序设计》**

**（第二版）**

**教案**

**北京出版社**

**课时分配表**

|  |  |  |  |
| --- | --- | --- | --- |
| **章序** | **课程内容** | **课时** | **备注** |
| **1** | **认识 C 程序** | **6** |  |
| **2** | **C 程序设计基础** | **7** |  |
| **3** | **顺序结构程序设计** | **5** |  |
| **4** | **选择结构程序设计** | **3** |  |
| **5** | **循环结构程序设计** | **6** |  |
| **6** | **数组** | **5** |  |
| **7** | **用户自定义函数** | **7** |  |
| **8** | **编译预处理** | **5** |  |
| **9** | **指针** | **9** |  |
| **10** | **构造数据类型** | **5** |  |
| **11** | **文件读写技术** | **6** |  |
| **12** | **C 语言综合实训** | **4** |  |
| **13** | **附录** | **4** |  |
| **总计** |  | **72** |  |

**第6课 数组**

|  |  |  |
| --- | --- | --- |
| **课 题** | 数组 | |
| **课 时** | 5课时（225 min）。 | |
| **教学目标** | **知识技能目标：**  1．了解数组的引入。  2．通过学习与练习掌握一维数组。  **思政育人目标：**  让学生通过学习数组，能从数组下标越界的危害中感悟“做事要有原则，做人要有底线”的哲理，养成细心、周密、诚信的软件服务意识。 | |
| **教学重难点** | **教学重点：**数组的引入  **教学难点：**二维数组 | |
| **教学方法** | 讲授法、问答法、讨论法 | |
| **教学用具** | 电脑、投影仪、多媒体课件、教材 | |
| **教学设计** | 第1节课：考勤（2min）--知识讲解（40min）--作业布置（3min）  第2节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第3节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第4节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第5节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min） | |
| **教学过程** | **主 要 教 学 内 容 及 步 骤** | **设计意图** |
| **考勤**  **（2min）** | ■【教师】清点上课人数，记录好考勤  ■【学生】班干部报请假人员及原因 | 培养学生的组织纪律性,掌握学生的出勤情况 |
| **知识讲解**  （40min） | **【教师】**数组的引入  我们来看下面的一个例子。  【例 6-1】输入 6 个数，将它们按输入次序的逆序输出。  源程序：  #include <stdio.h>  void main()  {  int x1,x2,x3,x4,x5,x6;  printf(" 请输入 6 个数 :");  scanf("%d,%d,%d,%d,%d,%d",&x1,&x2,&x3,&x4,&x5,&x6);  printf(" 逆序输出这 6 个数 :");  printf("%d,%d,%d,%d,%d,%d",x6,x5,x4,x3,x2,x1);  }  可以想象，若要逆序输出 100 个甚至更多个数据，该程序将会变成什么样子。  数组就是为解决这一类问题而提出的。  数组，顾名思义，是相同类型的数据组成的一个组。它用一组连续的内存空间，来存储一组具有相同类型的数据。也就是说，把相同类型的一系列数据统一编制到某一个组别中，这样就可以通过数组名 + 索引号简单快捷地操作大量数据。使用数组解决问题一般分为 4 个步骤：  （1）定义数组；  （2）给数组赋值；  （3）处理数组；  （4）输出数组。  其中，给数组赋值、处理数组和输出数组元素一般都要用到循环。  **【学生】**思考、讨论。 | **展示数组的引入，让学生更加仔细的阅读，从而激发学生的学习欲望。** |
| **作业布置**（3min） | **【教师】**布置课后作业  简述数组解决问题的步骤。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**一维数组  **任务描述**  求 Fibonacci 数列：1，1，2，3，5，8，…的前 20 个数。  **任务分析**  这是一个典型的迭代运算。迭代法的基本思想是：不断用新值取代变量的旧值，或由旧值递推出变量的新值。该数列的特点是：前两个数均为 1，从第 3 个数开始，每一个数是其前两个数之和。即：  F1=1 (n=1)  F2=1 (n=1)  Fn=Fn-1+Fn-2 (n>=3)  **预备知识**  **一、一维数组的定义**  一维数组是后面只有一个下标的数组，其定义形式为：类型说明符 数组名 [ 常量表达式 ];  例如：int a[10]；  上面定义了一个一维数组，其中，a 为数组名，此数组有 10 个元素，每个数组元素的类型都是整型。  【说明】  （1）数组名的命名规则与变量名相同，遵循标识符的命名规则。  （2）数组名的后面是用中括号括起来的常量表达式，不能使用小括号，例如，下面的方法是错误的：  int a(10);  （3）常量表达式即数组元素的个数，例如，a[10] 中的 10 表示 a 数组有 10 个元素，这 10 个元素是：a[0]，a[1]，a[2]，a[3]，a[4]，a[5]，a[6]，a[7]，a[8]，a[9]。  【注意】不能使用数组元素 a[10]。  （4）常量表达式中可以包含直接常量和符号常量，不能包含变量。也就是说，C不允许对数组的大小作动态定义，即数组的大小不依赖于程序运行过程中的变量的值。  例如，下面定义数组的方法是错误的：  int n;  scanf("%d",&n);  int a[n];  …  **二、一维数组元素的引用**  数组必须先定义后使用。C 语言规定：只能逐个引用数组元素，而不能一次引用整个数组。  一维数组元素的引用格式：  数组名 [ 下标 ]  【说明】下标可以是整型常量或整型变量，也可以是整型表达式。若数组长度为len，则数组下标的取值范围是 0~len-1，0 为下限，len-1 为上限。注意，数组下标从 0开始计算。  【例 6-2】利用数组实现【例 6-1】的功能。  源程序：  #include <stdio.h>  void main()  { int i,a[6];  printf(" 请输入 6 个数 :") ;  for(i=0;i<=5;i++)  scanf("%d",&a[i]); /\* 数组元素引用 \*/  printf("\n 逆序输出这 6 个数 :") ;  for(i=5;i>=0;i--)  printf("%d ", a[i]); /\* 数组元素引用 \*/  }  程序运行结果：  请输入 6 个数 :0 1 2 3 4 5 ↙  逆序输出这 6 个数 : 5 4 3 2 1 0  **三、一维数组的初始化**  一维数组的初始化就是给一维数组赋初值，其实现形式是：  数据类型 数组名 [ 数组长度 ]={ 值 1, 值 2, 值 3,……, 值 n };  例如：  int a[5]={1,2,3,4,5};  各元素的值为：a[0]=1，a[1]=2，a[2]=3，a[3]=4，a[4]=5。  按照赋初值的完整程度，又可以分为 4 种初始化方法。  （1）完全赋初值。例如：  static int a[5]={1,2,3,4,5};  （2）部分赋初值。例如：  static int a[5]={1,2,3};  【说明】数组 a[5] 只有前 3 个元素被赋予给定的初值，剩余的数组元素被初始化为 0。  （3）不指定数组长度赋初值。例如：  static int a[ ]={1,2,3,4,5,6,7,8,9,10};  【说明】对数组也可以不指定长度，本例中系统根据提供的初值个数自动将该数组长度取为 10。  （4）不赋初值。例如：  static int a[5];  【说明】对静态数组和全局型的数组，如果不赋初值，系统对该数组的全部元素赋 0。所以本例中 a[0]=0，a[1]=0，a[2]=0，a[3]=0，a[4]=0。  **任务实施**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* 程序 : program6\_1.c \*/  /\* 功能 : 求 Fibonacci 数列 :1,1,2,3,5,8,…… 的前 20 个数。 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  void main()  {  int i;  int f[20]={1,1}; /\* 部分赋初值 \*/  for(i=2;i<20;i++)  f[i]=f[i-2]+f[i-1];  for(i=0;i<20;i++)  {  if(i%5==0) printf("\n"); /\* 控制一行输出 5 个数 \*/  printf("%-8d",f[i]);  }  }  **【学生】**思考、讨论。 | **通过教师讲解，了解一维数组的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了一维数组，让学生能从数组下标越界的危害中感悟“做事要有原则，做人要有底线”的哲理，养成细心、周密、诚信的软件服务意识。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | **【教师】**布置课后作业  定义数组，用来存放下列数据：  （1）100 个学生的年龄；  （2）50 个学生的成绩；  （3）26 个英文字母。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**二维数组  **任务描述**  有一个 3×4 的矩阵，编程求出其中最大的那个元素的值及其所在的行号和列号。  **任务分析**  （1）定义一个 3 行 4 列的二维数组，用来存放 3×4 的矩阵，二维数组数据的录入可以采用初始化的方法，也可以采用双重循环来实现。  （2）再利用一个双重循环完成二维数组中最大值的寻找。  流程图如6-1 所示。    **预备知识**  **一、二维数组的定义**  二维数组是数组名的后面有两个下标的数组，其定义形式为：  类型说明符 数组名 [ 常量表达式 1] [ 常量表达式 2]  例如：int a[2][3]；  它是一个 2 行 3 列的二维数组，数组名为 a，此数组有 2×3=6 个元素，这 6 个元素是：a[0][0]、a[0][1]、a[0][2]、a[1][0]、a[1][1]、a[1][2] 。每个数组元素的类型都是整型。  【说明】  （1）类型说明符、数组名、常量表达式的定义与一维数组一致。  （2）二维数组的每一维元素的下标也都是从 0 开始的。常量表达式 1 表示二维数组的行数，常量表达式 2 表示二维数组的列数。二维数组的长度由该数组中两个中括号内的常量表达式的乘积决定。  （3）二维数组可以看成是类型为数组的一维数组，它的每个元素为一个一维数组。例如，上面定义的数组 a 可看作是一个一维数组，它有 2 个元素 a[0]、a[1]，每个元素又是一个包含 3 个 int 型元素的一维数组。  （4）在 C 语言中，二维数组在内存中的存放顺序是按行排列，占有一块连续的内存单元。即在内存中先存放第一行的元素，然后再存放第二行的元素，依次类推。  （5）定义二维数组时，不能用变量定义数组的行数和列数，行数和列数必须分别用一对中括号括起来。  例如，以下定义是不正确的：  int n;  int a[n][4];  多维数组的定义类似，例如：  int a[2][3][4];  定义了一个三维的 int 型数组 a，共包含 2×3×4=24 个元素。多维数组在内存中存放时，数组元素第一维的下标（最左边）变化最慢，最右边的下标变化最快。  **二、二维数组元素的引用**  二维数组元素的引用格式如下：  数组名 [ 行下标 ][ 列下标 ] 通常，二维数组的赋值和输出要使用双重循环，外循环控制行，内循环控制列。  例如：  int i,a[3][4]; /\* 定义二维数组 \*/  for(i=0;i<=2;i++)  for(j=0;j<=3;j++)  scanf("%d",&a[i][j]); /\* 二维数组元素赋值 ( 数组元素引用 ) \*/  【说明】下标可以是整型常量或整型变量，还可以是整型表达式。例如，a[2][3]，a[2-1][2\*3] 都是合法的。  在引用数组元素时，要注意下标不能越界，行下标的取值范围为：0~ 行下标 -1，列下标的取值范围为：0 ~ 列下标 -1。  **三、二维数组的初始化**  二维数组的初始化就是在定义二维数组的同时，给各数组元素赋初值，二维数组可以按行分段赋初值，也可以按行连续赋初值。  （1）按行分段赋值。例如：  static int a[2][3]={ {1,2,3},{4,5,6} };  第一个大括号里的数据依次赋给第一行的元素，第二个大括号里的数据依次赋给第二行的元素。  （2）连续赋值。例如：  static int a[2][3]={ 1,2,3,4,5,6 };  这种赋值方法是将所有的初值写在一个大括号内，按数组元素在存储时的排列顺序赋初值，存放结果与第一种相同。  （3）部分赋初值。例如：  static int a[3][3]={ {1},{2},{3} };  对每一行的第一列元素赋值，未赋值的元素取 0 值。  以上定义等价于 int a[3][3]={ {1，0，0}，{2，0，0}，{3，0，0} }；  （4）省略维数赋值。例如：  static int a[3][3]={ 1,2,3,4,5,6,7,8,9 };  可以写为：  static int a[ ][3]={ 1,2,3,4,5,6,7,8,9 };  【注意】  （1）只有对数组的全部元素赋初值时，数组的第一维长度可以省略，但第二维不能省略；  （2）若对数组的部分元素赋初值，则数组的第一维和第二维长度均不能省略。  【例 6-3】设计一个程序，输出下列矩阵。  程序分析：  （1）定义一个二维数组，用来存放 3×3 的矩阵，二维数组数据的输入可以采用初始化的方法，也可以采用双重循环来实现（本例中采用初始化的方法）。  （2）程序用一个双重 for 循环来输出矩阵，注意换行符的使用。  源程序：  #include <stdio.h>  void main()  {  static int a[3][3]={ {1,2,3},{4,5,6},{7,8,9} };/\* 二维数组初始化 \*/  int i,j;  printf(" 输出数组 a:\n");  for (i=0;i<3;i++)  {  for(j=0;j<3;j++)  printf("%3d",a[i][j] );  printf("\n"); /\* 每输完一行后换行 \*/  }  }  **任务实施**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* 程序 : program6\_2.c \*/  /\* 功能 : 求 3×4 矩阵的最大值及其所在行号和列号 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  void main()  {  int a[3][4],i,j,row,column,max;  printf(" 请输入 12 个整数 , 用空格隔开 :\n");  for(i=0;i<=2;i++) /\* 利用双重循环给二维数组赋值 \*/  for(j=0;j<=3;j++)  scanf("%d",&a[i][j]);  max=a[0][0];  row=0;  column=0;  for (i=0;i<=2;i++)  for(j=0;j<=3;j++)  if(a[i][j]>max)  { max=a[i][j];  row=i;  column=j;  }  printf("max=%d,row=%d,column=%d\n",max,row,column);  }  程序运行结果：  请输入 12 个整数 , 用空格隔开 :  1 3 -4 6 2 7 5 8 9 0 -11 -9  max=9,row=2,column=0  **【学生】**思考、讨论。 | **教师通过二维数组展示，让学生了解二维数组的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了二维数组，能够注重细节，一丝不苟，做事精益求精，具备从实践中找到答案的能力。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **定义一个二维数组，用来存放 60 个学生 7 门课的成绩。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**字符数组与字符串  **任务描述**  判断一个字符串是否为回文。回文是指顺读和倒读都是一样的字符串。例如：“aacdcaa”。  **任务分析**  根据回文的概念，判断字符串是否回文的方法是：先比较第一个字符和最后一个字符，若相同，则继续比较第二个字符和倒数第二个字符……，直到比较到中间字符为止。  如果每组字符的比较都相同，说明是回文；在比较的过程中，只要有一组比较不相同，则停止比较，说明不是回文。步骤如下：  （1）定义一个字符型数组，用来存储该字符串。  （2）设置两个整型变量 i 和 j，分别指向该字符串的首尾字符。  （3）循环，当 i<j 时，  比较 i 和 j 位置上的字符，若不相等，结束循环，执行（4）若相等，则 i 后移一位，j 前移一位，转向（3）  （4）若 i>= j，是回文，否则不是回文。  **预备知识**  **一、字符数组的定义及初始化**  字符数组的定义、初始化及引用方式与一维数组和二维数组基本一样，唯一的区别是，字符数组的数组元素的类型为字符型。  【例 6-4】字符数组的定义及初始化示例  (1)char c[10] ; /\* c 为一维的字符数组 \*/  (2)char str[5][10]; /\* str 为二维的字符数组 \*/  (3)char c[6]={"China"};  (4)char c[6]={'C',' h','i','n','a','\0'};  (5)char c[6]="China";  (6)char c[]="China"; /\* c 数组大小为 6 \*/  【说明】（3）~（6）等价，都是对字符数组 c 初始化，初始化完成后，c 数组元 素 的 值 依 次 为：c[0]=‘C’，c[1]=‘h’，c[2]=‘i’，c[3]=‘n’，c[4]=‘a’，c[5]=‘\0’。  **二、字符串**  1. 字符串和字符串结束标志  字符串是指若干有效字符的序列。C 语言中只有字符串常量，没有字符串变量，字符串不是存放在一个变量中，而是存放在一个字符数组里。  字符串作为一个整体，它是以“\0”作为结束标志的。“\0”是指 ASCII 码为 0 的字符，这个字符不是普通的可显示的字符，而是一个空操作符，它不进行任何操作，只是作为一个标记。  了解了字符串的含义，即可明确在不同的初始化方法下数据存放的区别。  例如：  char c[ ]={'C',' ','p','r','o','g','r','a','m'};  char c[ ]="C program";  这两种初始化方式的区别在于，末尾是否有一个字符串结束标志“\0”，这样不仅会导致两个字符数组有一个元素不同，还会导致字符数组长度不同。  2. 字符数组的输入和输出  字符数组的输入和输出有两种方法。  （1）逐个字符输入输出  在 scanf 函数和 printf 函数中，用格式符“%c”控  制输入或输出一个字符。  例如：  用 scanf( ) 输入：  char a[10];  for(i=0;i<9;i++)  scanf("%c",&a[i]);  用 printf( ) 输出：  char a[10];  for(i=0;i<9;i++)  printf("%c",a[i]);  （2）整个字符串输入输出  在 scanf 函数和 printf 函数中，用格式符“%s”控制输入或输出一个字符串。  例如：  用 scanf( ) 输入：  char a[10];  scanf("%s",a); /\* 数组名 a 的前面无 & \*/  用 printf( ) 输出：  char c[ ]="C program ";  printf("%s",c );  【说明】  （1）在 scanf 函数和 printf 函数中使用格式符“%s”输入输出字符串时，使用数组名而不是数组元素名。并且在 scanf 函数中，数组名前不加“&”，因为数组名本身就代表数组的起始地址。  （2）在输出字符串时使用格式符“%s”，输出的字符串是以“\0”结尾，如果数组长度大于字符串的实际长度，也只输出到“\0”为止。  （3）如果一个字符数组中包含一个以上的“\0”，则遇到第一个“\0”时就结束。  **三、字符串处理函数**  C 语言提供了丰富的字符串处理函数，大致可分为字符串的输入、输出、合并、修改、比较、转换、复制等几类。使用这些函数可以大大减轻编程的负担。用于输入输出的字符串函数，在使用前应包含头文件 stdio.h，使用其他字符串函数则应包含头文件 string.h。  下面介绍几个最常用的字符串函数。  1. 字符串输出函数 puts  格式：puts（字符数组名）  功能：将一个字符串输出到终端，即在屏幕上显示该字符串。  例如：  字符串处理函数  char s[ ]= "Good Morning!";  puts(s);  输出结果为：  Good Morning!  输出字符串时，字符串结束标志“\0”转换为换行符“\n”，即输出字符串后换行。  2. 字符串输入函数 gets  格式：gets（字符数组名）  功能：从键盘上输入 1 个字符串（以回车结束），存入指定的字符数组中。本函数得到一个函数值，即为该字符数组的首地址。  例如：  char s[20];  gets(s);  3. 字符串拷贝函数 strcpy  格式：strcpy（字符数组名 1，字符数组名 2）  功能：将字符数组 2 中的字符串拷贝到字符数组 1 中，字符串结束标志“\0”也一同被拷贝。字符数组 2 也可以是一个字符串常量。  例如：  char str1[20],str2[]="Good Morning!";  strcpy(str1,str2);  puts(str1);  输出结果为：  Good Morning!  使用 strcpy 函数，应注意以下几个问题。  （1）字符数组 1 应有足够的长度来容纳复制过来的字符串。  （2）字符数组 1 必须为字符数组名，字符数组 2 可以是字符数组名，也可以是字符串常量。  （3）C 语言不允许用下列方式将一个字符串常量或字符数组赋给一个字符数组：  char str1[20], str2[]="Good Morning!";  str1=str2;/\* 这种方法是错误的 \*/  （4）可以只将一个字符串的前面的 n 个字符拷贝到字符数组中。  例如：  strcpy(str1,str2,3);  其功能是将字符串 str2 前面的 3 个字符拷贝到字符数组 str1 中。  4. 字符串连接函数 strcat  格式：strcat（字符数组名 1，字符数组名 2）  功能：将字符数组 2 中的字符串连接到字符数组 1 中的字符串的后面，并删去字符数组 1 中字符串后面的串结束标志“\0”。字符数组 2 也可以是一个字符串常量。  【注意】字符数组 1 必须足够大，以便容纳连接后的字符串。  例如：  char str1[20] ="Good " ;  char str2[10] ="Morning!" ;  strcat(str1,str2);  上述程序段运行后，字符数组 str1 中的内容为：  Good Morning!  5. 字符串比较函数 strcmp  格式：strcmp（字符数组名 1，字符数组名 2）  功能：按照 ASCII 码顺序比较两个数组中的字符串，并由函数返回值返回比较结果。  比较规则是：从两个字符串的第一个字符开始比较，直到出现不同的字符或遇到 “\0”为止。如果字符都相同，则认为两个字符串相等；如果出现不相等的字符，则以第一个不相同的字符的比较结果为准。  字符串 1= 字符串 2，返回值 =0；  字符串 1> 字符串 2，返回值为一正数；  字符串 1< 字符串 2，返回值为一负数 ；  例如：  char str1[20] ="aabbbcc" ;  char str2[20] ="abcabc" ;  if( strcmp(str1,str2)>0) printf("str1>str2");  else if ( strcmp(str1,str2)==0 ) printf("str1==str2");  else printf("str1<str2");  6. 测试字符串长度函数 strlen  格式：strlen（字符数组名）  功能：测试字符串的实际长度（不包括字符串结束标志“\0”），并作为函数返回值返回。  例如：  char str[10]={ "China"};  printf("%d",strlen(str));  输出结果不是 10，也不是 6，而是 5。  7. 字符串大小写字母相互转换函数 strlwr 和 strupr  格式：  （1）strlwr（字符串）  （2）strupr（字符串）  功能：  （1）将字符串中的大写字母转换成小写字母。  （2）将字符串中的小写字母转换成大写字母。  **任务实施**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* 程序 : program6\_3.c \*/  /\* 功能 : 判断一个字符串是否回文 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <string.h>  void main()  {  char str[20];  int i,j;  printf(" 请输入需要判断的字符串 :");  gets(str);  i=0;  j=strlen(str); /\* 计算字符串的长度 , 注意数组长度与字符串长度的区别 \*/  while(i<j)  {  if(str[i]!=str[j-1]) break; /\* 此处为 j-1, 而不是 j \*/  i++;  j--;  }  if(i>=j) printf(" 该字符串是回文 !\n");  else printf(" 该字符串不是回文 !\n");  }  程序运行结果：  请输入需要判断的字符串 :aabbaa ↙  该字符串是回文 !  **【学生】**思考、讨论。 | **教师通过字符数组与字符串展示，让学生了解字符数组与字符串的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了字符数组与字符串，让学生领悟“纸上得来终觉浅，绝知此事要躬行”的含义，培养实践能力。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **编程，用 gets 函数输入一个字符串，然后用 puts( ) 函数输出它。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**冒泡法  用冒泡法对 10 个整数从小到大排列。  **程序分析**：  冒泡排序的方法是：先对 n 个数的每相邻的两个数进行比较，小数放在前面，大数放在后面，经过第一遍扫描后，最大的数被放在了最后；接着对前 n-1 个数进行同样的比较，将次大数放在倒数第二的位置上，以此类推，直到排序结束为止。在这个过程中，大数不断往下沉，小数不断往上冒，所以叫冒泡排序。例如：当 n=6 时，第一趟排序过程如图 6-3 所示，排序总过程如图 6-4 所示。    在本例中，将数组大小定义为 11，不用元素 a[0]，只使用 a[1] 到 a[10] 以符合人们的习惯。  源程序：  #include <stdio.h>  void main()  {  int a[11];  int i,j,t;  printf(" 请输入 10 个整数 ( 数据之间用空格隔开 ):\n");  for(i=1;i<=10;i++)  scanf("%d",&a[i]);  printf("\n");  for(i=1;i<=9;i++)  for(j=1;j<=10-i;j++)  if(a[j]>a[j+1])  { t=a[j];a[j]=a[j+1];a[j+1]=t;}  printf(" 从小到大排好序的数列为 :\n");  for(i=1;i<=10;i++)  printf("%d ",a[i]);  }  程序运行结果：  请输入10 个数（数据之间用空格隔开）：  7 6 0 12 -4 17 66 23 111 99 ↙  从小到大排好序的数列为：  -4 0 6 7 12 17 23 66 99 111  **【学生】**思考、讨论。 | **教师通过冒泡法展示，让学生了解冒泡法的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了冒泡法，让学生能够熟练地运用冒泡法排序。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  理解数组的概念，会根据具体的问题判断是否使用到数组。能够正确定义  数组，并引用数组元素。能用数组解决实际问题。 | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 教师既要注重学习任务的设计与布置，也要注重学习的主动性与思维能力的培养。既要重视学习结果，也要重视学习过程。 | |